直径

2019年3月22日 13:35

-
- 树的直径

一. 树的直径

1. 直径/最长链：两结点间最大距离（边权和）（所在的路？）
2. 树形dp法：
   i. 有向树的调用时必须从根节点开始调用，且每条边必须是从父指向子，而无向树（或者说每条边都正反存两次）就无所谓从谁调，但有向图必须通过v[y]continue来避免回路
   ii. 因为树是n点n-1边，因此可以用v[]剪枝，时间复杂度是O（n）
   iii. 从指定根出发的两条路的和可能有重复边，但指定根的两不同儿子出发的路，加上根到儿子的距离肯定没有重复边
   iv. 先用D[x]存x为根，指向叶的最长路径D[x]=max(D[y])+w[x][y]
   v. 再用F[x]存经过x的最长路径F[x]=max(D[yi]+w[x][yi]+D[yj]+w[x][yj])
   vi. 令j>i，则遍历到子节点yj时已经有D[x]=max(D[yi]+w[x][yi])，因此实际只需一层y循环，F[x]=max(F[x],D[x]+D[yj]+w[x][yj])
   vii. 为防止重复计数F[x]=D[x]+D[x]，先更新F[x]再更新D[x]
   viii. 最后需要求的直径diam是maxF[x]，其实不需要存每个F[x]
   ix.
```
void dp(int x){   //更新d[x]和经过x的最长diam
    v[x]=1;
    for(int i=head[x]; i; i=next[i]){
        int y=ter[i];
        if(v[y])
            continue;
        dp(y);  //先更新好子节点
        diam=max(diam,d[x]+d[y]+edge[i]);
        d[x]=max(d[x],d[y]+edge[i]);}}
```
3. 两次搜索法（dfs或bfs）
   i. 先从任意点出发搜出最远的一点，该点一定是直径的某一端
   ii. 从这一端出发，搜到最远的一点是直径另一端
   iii. 自不量力的尝试证明一下：
   iv. 1出发点是端点，最远点是直径另一端，显然成立
   v. 2出发点不是端点，假设最远点不是端点，设最远点距离d
   vi. 再设到相隔较远的端点的距离是D，则D<d
   vii. 则从另一端点到d对应点的距离肯定>到D点的距离，与假设矛盾
   viii. 搜索参数至少是当前起点和当前距离，为避免重复，可用v数组判断提前continue，或根据只有父节点==与该结点相邻且已搜索过的结点，将父节点编号作为函数参数传入递归搜索
   ix.
```
vector<int>to[MN];
int ans;
int src1,src2;
```

```
void dfs(int f=1,int x=1,int dis=0){
        if(dis>ans)
                ans = dis,
                src1 = x;
        for(auto y : to[x])
                if(y!=f)  //是父节点 iff 已遍历过
                        dfs(x,y,dis+1);}
x.  for_(i,1,n)
                scanf("%d%d",&a,&b),
                to[a].push_back(b),
                to[b].push_back(a);
        dfs();
        dfs(src1,src1);
```

xi.